CMPSC-265
Data Structures and Algorithm

Zaihan Yang
zyang13@suffolk.edu

Department of Math and Computer Science
Suffolk University

Fall 2019

About me

- Prof. Zaihan Yang
- PhD in Computer Science from Lehigh University
- Email: zyang13@suffolk.edu
- Office: 73 Tremont, Room 8098
- Office hour:
  - Wed: 1.45pm-2.45pm
  - Fri: 12.30pm-1.15pm; 3pm-4.30pm

CS265 Course Overview

- Data Structure: arrangements of data in a computer.
- Algorithm: methods of manipulating the data in data structures to solve problems.
- Using Java to implement and run application over data structures and algorithms.

CS265 Course Overview

- We will basically cover 15 chapters of the textbook.
- Topics (not limited):

Data Structure
Arrays
Stack
Queue
Priority Queue
Linked list
Tree (binary tree, binary search tree, red-black tree)
Binary Heap
Hash table
Graph

Algorithm
Sorting: insertion sort, selection sort, bubble sort, merge sort, quick sort, heap sort, topology sort
Searching: linear search, binary search, hashing.
Algorithm on Tree: traverse trees, and etc
Algorithm on Graph: Breadth first search, depth first search, minimum spanning tree, finding shortest path

CS265 Course Overview

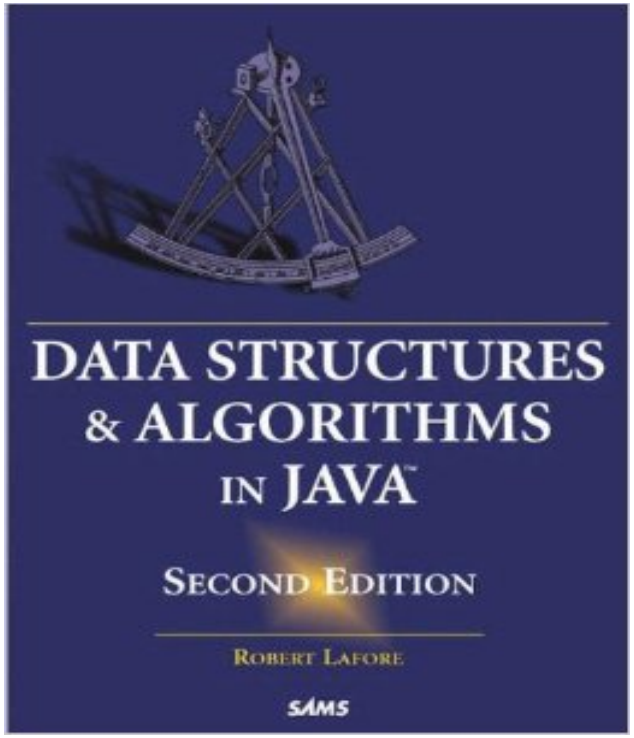
- Course Objectives
  - Grasp the most important data structure and algorithms in computing.
  - Know how to choose/design data structures and algorithms as programs to solve real world problems
  - Know how to analyze complexity/efficiency of an algorithm
  - Get programming skills improved via hands-on programming practice.

CS265 Course Overview

- Data structure and algorithm are very important in modern computing.
- Their impact is board and far-reaching.
- They are the core and fundamental for all upper-layer computer science research and directions.
- Key knowledge you should know and will be tested on job interviews as a software engineer

CS265 Course Overview

- Prerequisite:
  - CMPSC 132 (Computer Science II).
- Textbooks



- Data Structures and Algorithms in Java.
- By Robert Lafore
- The 2nd version
- Publisher: SAMS
- You can buy it on Amazon.
- I have also posted the electronic version on Blackboard.

CS265 Course Organization

- Class meetings: MW 12.15pm-1.30pm (75 minutes)
  - I will do most of the lecturing and codes demoing in class.
  - You might have in-class ungraded quizzes the last ~10 minutes for knowledge refreshment.
  - Roughly every two weeks, you will have in-class graded quizzes (the last ~15 minutes).
  - Roughly every two weeks, you will have in-class discussion on HWs (the last ~15 minutes)
- Blackboard: is used for course management and HWs submission.

CS265 Course Organization

- There are two types of assignments that you have to submit:
  - Weekly Homework Assignments (Starting from the 1st week)
  - Bi-Weekly in-class quizzes (Starting from 3nd week) (on general)
- Exams:
  - Two mid-terms (75 minutes) and one final exam (150 minutes)
  - Written exams.
  - Closed books and all electronic devices closed

CS265 Course Organization

- Homework
  - You will have weekly programming homework.
  - Homework normally will be due on every Sunday night 11.59pm.
  - You are expected to finish all HWs independently on your own. You need to strictly follow the instructions and submit the program in proper style.
- Bi-Weekly Quizzes (On general):
  - You will take them on Wednesday's class.
  - Most of the problems would be conceptual level understanding problems, like multiple choices, true/false, and short answers to questions.
- Weekly reading assignments

CS265 Grading Policy

- Late policy: 10 points off per day late
- All homework and exams are subject to the honor code. Plagiarism in any form will not be tolerated.
  - Homework: 40%
  - Class participation: 5%
  - Quizzes: 5%
  - Mid-term Exam 1: 15%
  - Mid-term Exam 2: 15%
  - Final Exam: 20%

CS265 Grading Policy

- Final grades will be computed based on the following standard scale:

A	93-100
A-	90-92
B+	86-89
B	83-85
B-	80-82
C+	76-79
C	73-75
C-	70-72
D+	66-69
D	63-65
D-	60-62
F	<60

Java

- We use Java in this course
- The concepts are independent of languages
- I assume prior familiarity with Java
  - Review the basics:
    - How to write and run a Java program
    - Object-oriented basics (defining a class, constructor,…)
- Object-oriented (Vs Procedural)
  - Reusability, efficiency,…
  - Object: a repository of data.
  - Class: a type of object. Many objects of the same class might exist.
  - Method: A function that operates on an object or a class.

Hello World in Java

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

- javac HelloWorld.java: // Compile: converts to bytecode object file HelloWorld.class
- java HelloWorld // interpret, Run

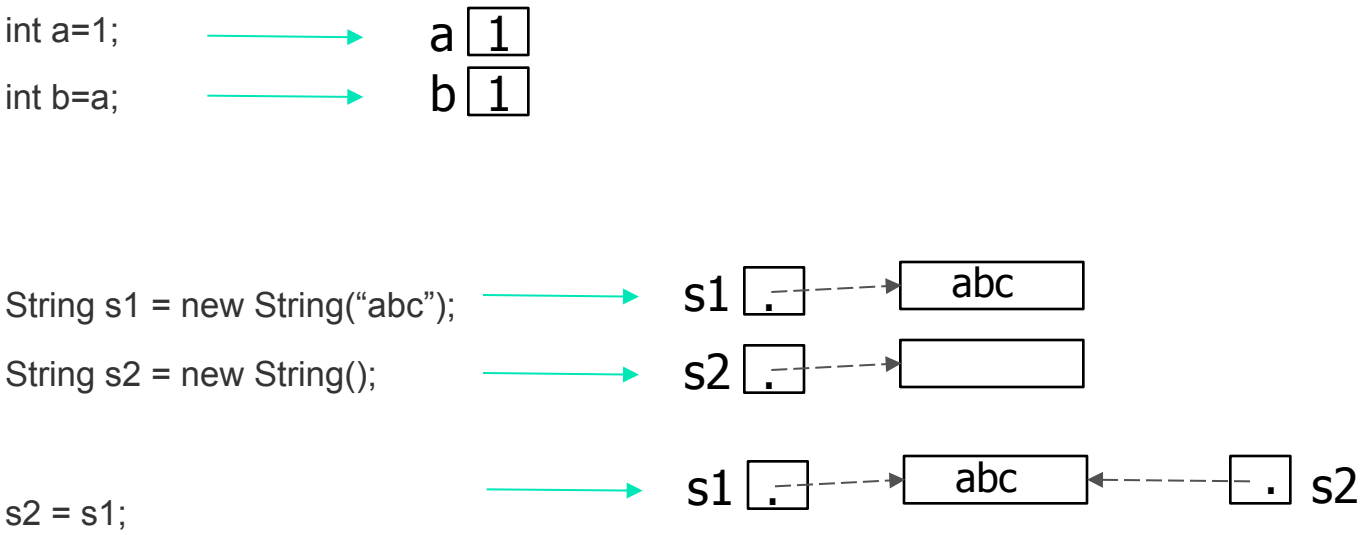
Java Basics

- Each Java source file should contain at least one class, and the name of the source file is just the class name with .java extension.
- A Java program is made up of one or more source files, each containing at least one class.
- Java is case-sensitive.
- A code block in Java is indicated by a {}
- Method is the basic unit of work in Java program that executes a specific task.
- A method has a method name; It can return a value of a specified return type or returns nothing (void)
- A method can have zero, one, or more than one parameters.
- We can call a method by its name and pass in arguments to its parameters.
- A Java program needs to have a main method from which to get started.

Java Basics: Basic terms and components

- **Data type:** Each data format is called a data type.
  - Java is a type-safe and strongly-typed language, you should declare the data type for any constants, variables, method return values as well as method parameters.
  - There are 8 primitive data types in Java: int, char, boolean, double, float, byte, long, short. (A single value is stored).
  - Reference type (Strings, other objects), where the reference is stored.
- **Variable:** A variable is a place in memory that holds a single value, and has a name and a data type.
  - The data type of a variable has to be declared: DATA_TYPE VARIABLE_NAME, i.e., int a;
  - We can initialize and assign a value to a variable, i.e. int a = 10;

Data types in Java (example)



s1==s2 vs s1.equals(s2)

String s1 = "abc";

String s2 = "abc";

if (s1 == s2) . . . // returns false

if (s1.equals(s2)) … // returns true

Type Casting

- Type casting converts a value of one type to a value of another type
  - *Implicit*
  - *explicit*

```
int i = 5;

//implicit
double d = i;

//explicit
int j = (int)d;
```

Basic terms and components

- **Identifier:** an identifier is a name of something like variables, constants, methods or classes.
  - A valid identifier is made up of letters and digits, and two special characters ($ and _), and it cannot start with a digit.

- **Constant**: A constant is a special variable whose value can never change.
  - We can declare a constant like: final int a=10, or a class constant like: public static final int a=10;

- **Literal**: a literal is a value you type directly into the code.
  - An integer literal: 0, 1, -1;
  - A decimal literal: 2.2, -1.5, 0.3
  - A char literal: ‘a’, ‘z’, ‘ ‘ ;
  - A String literal: “hello”, “apple”, “”, “ “

Basic terms and components

- **Expression**: an expression is anything that the compiler can turn into a value.
  - There are five types of expressions: literals, variables, contants, calculations, and method calls.
- **Statement:** a line of code the represents a complete command.
  - Each Java statement should end up with a semicolon ;
  - The print-out statement is: System.out.println(EXPRESSION);
    System.out.prin(EXPRESSION);
- **Scope**: the scope of a variable or constant is from where it is declared to the closing curly brace of the code block.

Operators

Types	Operators
Arithmetic operators	Unary operators: ++, --, +, - (this is for sign) Additive operators: +, - Multiplicative operators: *, /, %
Assignment operators	=, +=, -=, *=, /=, %=
Relational operators	==, != (equality operators) <, >, <=, >=
Logical operators	&&, ||, !

Operators' Precedence level (higher to lower)

Types	Operators
Unary operators	++, --, +, -
Multiplicative operators	*, /, %
Additive operators	+, -
Relational operators	<. >, <=, >=
Equality operators	==, !=
Logical operators	&&, ||, !
Assignment operators	=, !=, -=, *=, /=, %=

Control Structures

- Boolean expression evaluates to the values true or false

- Control structures:
  - Decisions:
    - if, if-else, switch
  - Loops
    - for, while, do-while

Control Structures

- ***for*-loop**: can help us to do the same thing multiple times.
- A *for* loop statement has two major parts
  - Three statements inside parentheses that control the loop
  - A series of statements executed with each iteration through the loop; These series of statements are called the loop body and are enclosed in {}
- The format of a *for*-loop

```
for ( INITIALIZATION;  CONTINUATION_TEST;  UPDATE )  {
    STATEMENT;
    ...
}
```

```
for (int i=1; i<=10; i++){
  System.out.println("hello");
}
```

- *for*-loop can be nested.
- ***if/else* statement or switch:** can help us make selections. It allows a section of code to be executed when some condition is true and some other code to be executed when the condition is false.
- The basic format:

```
if (TEST_EXPRESSION) {
    STATEMENT;
    ...
}
```

```
if (TEST_EXPRESSION)  {
    STATEMENT;
    ...
} else {
    STATEMENT;
    ...
}
```

- If/else statement can also be nested.

Control Structures

- while-loop and do-while-loop: both of them can be used for indefinite loops, whose number of iterations is not known before codes running.

```
while ( TEST_EXPRESSION ) {
   STATEMENT;
   ...
}
```

```
do {
   STATEMENT;
   ...
} while (TEST_EXPRESSION) ;
```

- The do-while loop body will at least be executed once since the test-expression is evaluated after the loop body.
- If it is used for definite loops whose number of iterations is known, while-loop is functionally equivalent to for-loop, and we can rephrase between them.

Text Processing

- Processing characters and Strings
- *char* primitive data type
  - Encoding of char type: Unicode and ASCII encoding scheme: characters are represented internally as an integer;
    - Each character has one corresponding int value. (‘A’ is 65, ‘a’ is 97)
    - Comparison between char type variables ch1 and ch2:
      - ch1 is less than ch2 if it comes alphabetically before ch2
- String: a sequence of characters.
  - String s = “hello”;
  - int length = s.length();
  - char c = s.charAt(0);
  - String subStr = s.substring(0,3); String subStr2 = s.substring(3);

Standard Input/Output

- Can use the built-in Scanner class to get input from keyboard.
  - Scanner console = new Scanner(System.in);
  - int a = console.nextInt();
  - String s = console.nextLine();
- System.out.println() to output to standard output on screen.
  - System.out.println("Hello, World");

File Processing

- Read data from file
  - Scanner input = new Scanner(new File(“test.txt”));
  - String line = input.nextLine();
- Write data to a file
  - PrintStream output = new PrintStream(new File(“output.txt”));
  - output.println(“Hello!”);
- Can apply try/catch statement to handle exceptions.

Class and Object

- A class is a programmer-defined type.
  - It provides a template (blueprint) of the common attributes and behaviors that objects of this class will have.
  - The class behaviors/operations are implemented by class methods.
  - The class attributes (data items) are called fields.
- An Object is an instance of a class
  - Really occupy memory in the computer
  - All objects of a class have the same methods.
  - All objects of a class have the same attributes.
    - For different objects, each attribute can hold a different value.

Example (a class)

```
class BankAccount
{
        private double balance; // account balance
        public BankAccount(double openingBalance) // constructor
        {
                balance = openingBalance;
        }
        public void deposit(double amount) // makes deposit
        {
                balance = balance + amount;
        }
        public void withdraw(double amount) // makes withdrawal
        {
                balance = balance - amount;
        }
        public void display() // displays balance
        {
                System.out.println("balance=" + balance);
        }
} // end class BankAccount
```

Example (Objects)

- Creating Objects:
  - BankAccount ba1 = new BankAccount(100.00); // instantiate
- Accessing Object Methods:
  - ba1.display();
  - ba1.deposit(74.35);

Inheritance and Polymorphism

- Inheritance:
  - The creation of one class (called the *extended* or *derived* class) from another class (called the *base* class). The extended class has all the features of the base class, plus some additional fields and methods.
    - Example, a *secretary* class derived from a general *employee* may include a *typingSpeed* field that was not in the employee class.
- Polymorphism:
  - Having one method call work on different classes, even if those classes need different implementations of the method call.
    - Example, calling *display()* for a secretary object invokes a different implementation from calling *display()* for a manager class.

Practice

- Write a Java program that can read a string from keyboard. Call a method isPalindrome() to judge whether this String is a palindrome or not.
- A palindrome string is a string which reads the same forward and backward.
- i.e.: kayak

Sample Codes

```
public static boolean isPalindrome(String s){
    for (int i=0; i<s.length()/2; i++){
        if (s.charAt(i)!=s.charAt(s.length()-i-1)) return false;
    }
    return true;
}
```